CMPT 250
Midterm 1

55 minutes   Closed Book   Nov. 26, 1997
Marks

3   1. Robustness is a term used to describe a piece of software.
       To what does it refer?

20 2. Give the Eiffel code to define a descendant of
       BASIC_BINARY_TREE_UOS[G], called STR_TREE, to store strings.
       Each tree represents a string expression. Instances of
       STR_TREE are to be created by the standard *make* and
       set_values routines of BASIC_BINARY_TREE_UOS[G]. In addition
       to the inherited routines, it is to have one new STRING
       function called *value*. This function is to return the STRING
       value of the expression represented by the tree. The value
       of a tree is obtained as follows. The value of a leaf node
       is the string stored in the node. The value of an interior
       node storing the string "append" is the value of its right
       subtree appended to the end of the value of its left subtree.
       (Recall Eiffel STRINGs have a procedure called *append* that
       does this task.) The value of any other interior node should
       be just the empty string (independent of its subtrees).

       Thus the tree below has value "abcdpqr".
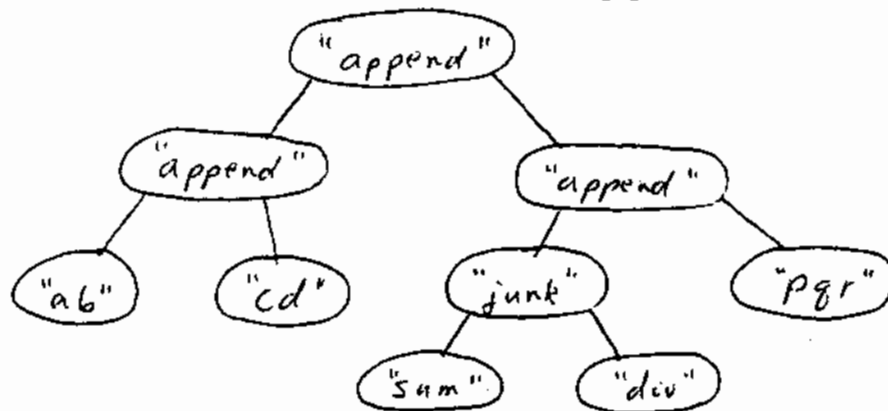
       For reference, the features of BASIC_BINARY_TREE_UOS[G] are:
       Note that to shorten the description, BB_TREE[G] is used
       instead of BASIC_BINARY_TREE_UOS[G].
          make
          set_values (lt:BB_TREE[G]; x:G; rt:BB_TREE[G])
          root_left_subtree : BB_TREE[G]
          set_root_left_subtree (newt:BB_TREE[G])
          root_right_subtree : BB_TREE[G]
          set_root_right_subtree (newt:BB_TREE[G])
          root_item : G
          set_root_item (x:G)
          insert_root (x:G)
          insert_root_left (x:G)
          insert_root_right (x:G)
          delete_root
          make_cursor : TREE_CURSOR_UOS [G]

                                                                       1

8   3. For <u>each</u> of the following routines, give its time
requirements using $O(.)$ or $\Theta(.)$ notation (or both), which
ever is appropriate. Express your final answer in as simple
a form as possible, without losing any accuracy.

Assume that
  s is a procedure with its time given by $T_s(n) \in O(\log_2 n)$.
  t is an INTEGER function with time requirements given by
$$T_t(m) \in \Theta(m).$$
  a is an array with dimension 1 to m
  b is an array with dimension 1 to n

Note that these routines are not meant to do anything useful
or interesting. Also, read the routines carefully so as not
to miss any routine calls.

```
d (x:INTEGER) : INTEGER is
   local i:INTEGER
   do
      from i:=1
      until i>m or else a.item(i) = x
      loop

          s

         i := i + 1
      end
      Result := i
   end

p is
   local j:INTEGER
   do
      from j := 1
      until j>n
      loop
         a.put( d(j) + t(j), j)
         j := j + 1
      end

      q(n)

   end

q (k:INTEGER) is
   do
      b.put( t(2*k+1), k)
      if k-1 > 0
      then
         q(k-1)
      end
   end
```
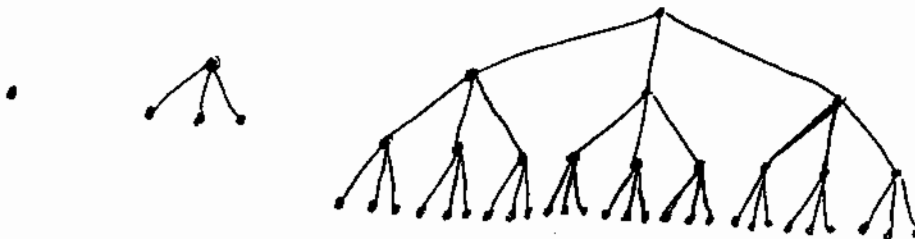
7 4. The question is to give the axiomatic definition for an
      operation on <u>ordered</u> binary trees. You should assume that
      make and *set_values* are the build operations with the usual
      parameters and semantics. The operation to be done is to be
      called *duplicate_root*. The semantics of the operation are to
      insert another copy of the root item into the tree. The
      duplicated copy can be inserted anywhere convenient as long
      as the tree is still ordered. Thus if initially a tree
      contained the value 12 three times, one of which was at the
      root, then after the operation *duplicate_root*, the tree
      should contain the value 12 four times. The number of
      occurrences all other items should not change.

      Give the signature, any pre-conditions, and the axioms.

12 5. Prove by induction that the number of nodes in a complete
      ternary tree of height h is $(3^{h+1}-1)/2$. Recall that the
      height of a tree is -1 for the empty tree, and otherwise the
      length of the longest path from the root to a leaf. Each
      node in a ternary tree has between 0 and 3 children. In a
      complete ternary tree each node has either 0 or 3 children
      (never 1 or 2). Also in a complete ternary tree, all leaves
      are the same distance from the root. Finally the empty tree
      is a complete ternary tree. The diagram below gives several
      complete ternary trees.